ADAPTIVE ALGEBRAIC SMOOTHERS*

BOBBY PHILIP † AND TIMOTHY P. CHARTIER ‡

Abstract. This paper will present a new method of adaptively constructing smoothers based on Local Sensitivity Analysis (LSA) for multigrid methods. The method can be used in the context of both geometric and algebraic multigrid methods. It is suitable for both constant and variable coefficient problems. Furthermore, the method can be applied to systems arising from both scalar and coupled system partial differential equations (PDEs), as well as linear systems not arising from PDEs. The simplicity of the method will allow it to be easily incorporated into existing multigrid codes while providing a powerful tool for adaptively constructing smoothers tuned to the problem.

Key words. Adaptive smoothers, block smoothers, line smoothers, multigrid, algebraic multigrid

AMS subject classifications. 65H10, 65F10, 65N50, 65N55

1. Introduction. This paper presents a new method that uses Local Sensitivity Analysis (LSA) to identify blocks of variables in a linear system that are strongly coupled. With such information regarding the coupling of variables, we construct stationary block iterative methods that can be used, for instance, as smoothers in multigrid methods. The smoothers so constructed can be used in the context of both geometric and algebraic multigrid methods. The method is suitable for both constant and variable coefficient problems. Furthermore, the method has been applied to systems arising from both scalar and coupled system partial differential equations (PDEs), as well as linear systems not arising from PDEs. The simplicity of the method will allow it to be easily incorporated into existing multigrid codes. Furthermore, it is possible to adaptively vary the size and strength of the blocks leading to the construction of a parametrized family of block iterative smoothers which can be tuned to the problem based on efficiency or convergence criteria.

1.1. Related work. Previously, much effort has been devoted to developing algorithms for matrix reorderings. Much of the motivation for this work came from research on direct solvers. Examples include the Nested Dissection algorithm [12], the multiple minimum degree algorithm [19], and independent set reorderings [18]. Similar ideas for matrix partitioning have also been used for enabling parallel processing of large linear systems [15]. The main focus in the case of direct solvers was to develop algorithms that minimize fill-in during matrix factorizations. In the case of parallel processing the focus was on identifying blocks of variables that were independent from each other and hence enabling asynchronous processing. Related work in the context of multilevel algebraic preconditioners [7, 24] report that the formation of dense diagonal blocks using the matrix reorderings appeared to have a beneficial effect on the convergence rate of block iterative methods. Further, [1] presents a block ordering method based on combinatorial considerations that formed dense variable sized diagonal blocks. Numerical tests presented there showed that the block iterative methods based on the reorderings performed better in general. Parter [22] provides an example of "multiline" iterative methods for systems arising from discretizations of elliptic partial difference equations (PDEs) and Varga [30] provides references to block iterative methods for elliptic difference equations and theory for such methods in the case of M-matrices and Stieljes matrices. In the context of multigrid methods Yavneh [31] and Brandt [5, 4] provide guidance for constructing multigrid smoothers for complicated PDE

*This work was supported under the auspices of the U.S. Department of Energy under DOE contract W-7405-ENG-36 at Los Alamos National Laboratory, an affirmative action/equal opportunity employer.

†Mathematical Modeling and Analysis Group, Los Alamos National Laboratory, Los Alamos, NM, 87545. *email*: bphilip@lanl.gov.

‡Department of Mathematics, Davidson College, Davidson, NC, 28035. *email*: tichartier@davidson.edu.

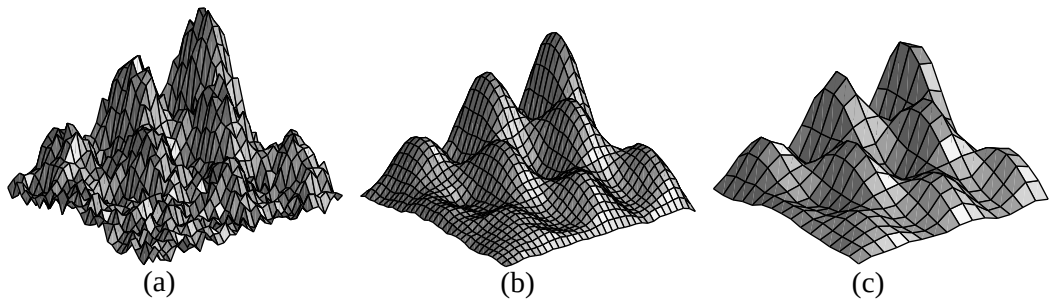


FIG. 2.1. *Error of an elliptic problem in 3 stages of a multigrid cycle. (a) Initial error on fine grid. (b) Smoothed error on fine grid after relaxation. (c) Smoothed error approximated by the coarse grid.*

systems. Smoothers based on Sparse Approximate Inverse techniques are developed in [26] and [8]. The methods described in this paper differ from the existing methods as they are based on sensitivity analysis and offer the ability to adaptively construct a whole family of smoothers that differ in convergence rates and efficiency. The smoothers developed in [26, 8] can be adapted, but this is done by altering the sparsity pattern and the degree of fill-in allowed, and not based on the formation of strongly connected blocks. Adaptivity of the smoothers is an important feature which is being utilized in ongoing research by the authors to construct fully adaptive multigrid methods. We note that recent work on energy-based coarsening [6] for algebraic multigrid methods is closely related and the criteria for coarsening proposed there can be derived as a special case of the general methodology presented here. Future work will address the feasibility of using the criteria developed in this paper for coarsening in adaptive AMG methods.

1.2. Outline. In the following section of this paper, a review of geometric and algebraic multigrid methods presents a broad context for the applicability of the methods of this paper for a multigrid process. Further, the section outlines how previous research in geometric methods has often required specially designed smoothers while research on algebraic methods have concentrated on altering the process of constructing coarse-grid components as smoothing is generally fixed to a simple relaxation scheme. Section 3 introduces the use of LSA for identifying strong coupling between variables. Section 3.7 introduces the adaptive algebraic smoother with Section 4 presenting numerical results for scalar and coupled system PDEs and to systems not generated from PDEs.

2. Review of Multigrid. Periodically in this paper, the potential role of adaptive algebraic smoothers in multigrid methods will be discussed. As such, we begin with a brief review of multigrid.

2.1. Geometric Multigrid (GMG). Multigrid methods solve discrete linear systems that often arise from discretizing PDEs. Specifically, such methods seek the solution $\mathbf{x} \in \Re^n$ to the linear system

$$A\mathbf{x} = \mathbf{f}, \tag{2.1}$$

where A is an $n \times n$ global matrix for a PDE. It is assumed in this paper that A is nonsingular, i.e., a unique solution, $\mathbf{x}$, to (2.1) exists. The multigrid iterative cycle begins with an initial guess $\mathbf{x}^0$ that yields the error $\mathbf{e}^0 = A^{-1}\mathbf{f} - \mathbf{x}^0$.

The first step toward this goal is to perform relaxation (or smoothing) on (2.1). However, after only a few steps of smoothing, continued iterations of relaxation would result in degradation of the convergence factor as only those components of the error that are not efficiently reduced by smoothing remain. Instead, GMG now attempts to correct the fine-grid approximation by a coarse-grid approximation to this error. This is done by first forming the *residual*, $\mathbf{r}^j = \mathbf{f} - A\mathbf{x}^j$, and posing the *residual equation*:

$$A\mathbf{e}^j = \mathbf{r}^j. \tag{2.2}$$

(Here j refers to the j-th iteration of the smoother.) GMG attempts now to solve the residual equation.

While we understandably expect $\mathbf{e}^0$ to contain both smooth and oscillatory components as seen in Figure 2.1 (a), carefully chosen relaxation schemes quickly dampen the oscillatory components of the error leaving only smooth components (Figure 2.1 (b)). As such, $\mathbf{e}^j$ can be represented on a coarser scale. This is done by forming the coarse-grid version of residual equation (2.2) and solving, resulting in a coarse-grid approximation to $\mathbf{e}^j$, as seen in Figure 2.1 (c). This coarse-grid approximation is interpolated to the fine grid and used to correct the approximation $\mathbf{x}^j$, effectively reducing the smooth components of the error. This is the basis of what is called *coarse-grid correction*.

The coarse-grid problem is not solved directly, but by a combination of smoothing and correction from still coarser grids. One full application of this recursive procedure constitutes a *multigrid cycle*. Well designed multigrid methods must sufficiently reduce all components of the error by a combination of fine-grid relaxation and coarse-grid correction. While sometimes separated for purposes of discussion, smoothing and coarse-grid correction are partners that combine to create the efficiency and power of the multigrid cycle: multigrid methods must be designed so that both mechanisms complement each other's efforts. A properly designed cycle reduces all error by a factor independent of the size of the fine grid problem, giving GMG its optimal solution time.

2.2. Algebraic multigrid. GMG methods are scalable for many regular-grid problems. Yet, they can be difficult to develop for the large unstructured grids that many simulations require. Algebraic Multigrid (AMG) attempts to overcome this difficulty by abstracting multigrid principles to an algebraic level so that the algorithm is more automatic and robust. AMG is effective on a large class of problems (see, e.g., [11, 23]), especially scalar elliptic partial differential equations. Still, important applications exist, many that lie in system PDEs and nonsymmetric problems, that are difficult for AMG to treat. It is important to note that iterative methods in general have failed to achieve full optimality for such problems.

AMG differs from GMG in that it attempts to choose components of the multigrid process automatically with only knowledge of the algebraic system. AMG methods typically fix the smoother, choosing for example Gauss-Seidel as the smoother on all grid levels. AMG then attempts to algebraically choose coarse-grid correction components [23, 13, 10] during a setup phase prior to the actual solution phase. In the context of AMG, smooth error need not necessarily be geometrically smooth. Instead, algebraically smooth error remaining after smoothing simply refers to error, $\mathbf{e}$, that is not significantly reduced after applying the smoother, S, i.e., $S\mathbf{e} \approx \mathbf{e}$ ([27], Appendix A).

2.3. A role for an automated, problem-dependent smoother in GMG and AMG. While the necessary components of a multigrid cycle are well-defined as detailed above, identifying how to build such components that will lead to an efficient iterative method is not always so clear. For example, other than for the simplest cases, it can be necessary to construct problem dependent components for smoothing and/or coarse-grid correction. GMG methods typically fix the hierarchy of coarse grids in advance, limiting the coarse-grid components that can be modified. Hence, in the design of a GMG solvers the burden shifts towards the design of smoothers that must eliminate oscillatory error components that fixed coarse-grid correction components cannot tackle.

In contrast, an AMG algorithm contains the setup phase which has the burden of producing a coarsening process that approximates error that the fixed smoothing scheme cannot reduce. Fixing the smoother has meant that in general AMG methods have trouble when used for solving coupled PDE systems that may have strong intervariable couplings due to failure of the smoother to converge, requiring the design of a problem dependent smoother by a multigrid expert. By automating the process of smoother selection we hope to shift some of the burden of an efficient AMG algorithm from coarse-grid correction to smoothing. While not addressed in this paper and the subject of future research, we note that our methods can potentially be also used to develop algorithms for the coarsening process in AMG.

This paper introduces a method to automatically create block smoothers, which refers to relaxation schemes where a set of variables are updated simultaneously in the relaxation scheme. It is possible to develop block versions of the Jacobi and Gauss-Seidel relaxation schemes as well as other

variants depending on the order of the updates of the blocks. Line smoothers are examples of block relaxation schemes which are effective when anisotropies align along grid directions for scalar problems. Alternating direction line smoothers are used when anisotropies do not align along grid directions. Segment smoothers refer to smoothers that form blocks on segments of lines. Nodal smoothers update a coupled set of variables at a node when discretizing a system PDE. Smoothers based on ILU are also possible. As is evident, a wide variety of smoothers for GMG methods that tackle different problem dependent characteristics have been developed. We refer to [27] for a more detailed description of these options and references. For complex geometries and unstructured grids constructing such smoothers is not easy. In general, knowledge and detailed analysis of the physical system by a multigrid expert has been required to determine an appropriate smoother other than for the simplest problems. Other than [31], we are unaware of methods that attempt to automatically construct a suitable smoother for a given problem, which is the goal of the methods introduced in this paper.

3. Determining Strong Coupling with Sensitivity Analysis. In this section, we outline how sensitivity analysis is used to identify strong coupling between variables in a linear system. Again, consider the linear system (2.1). Component-wise the solution can be written as:

$$x_i = \sum_{j=1}^{n} (A^{-1})_{ij} f_j \quad \text{for } i = 1, 2, \ldots n.$$

[**TODO**: Bobby, I changed this as I believe i is to be fixed which wasn't clear to me in the previous version of this equation. Further, the comment is made below that this makes it clear that f_j has a strong effect on x_i relative to the size of the corresponding element in the inverse. Doesn't this assume a geometrically smooth f? That is, isn't this assuming relatively close values in f? If f is highly oscillatory, then this might not be the case. This is similar motivation to how AMG justifies choosing strong connections from the row. However, AMG makes the point on u_j rather than f_j. Should we note something here? In a sense it is implicit in the comment but the conclusion, to me, is only clear upon such an assumption.Tim, I've got to think about this one. I'm not quite sure how to reword. As I understand it you are thinking of the cumulative effect of all the f_js on x_i. I am more thinking of a single value f_j. If f_j is to have any significant effect on x_i(quite independent from any of the other nodes), then the coefficient A^{-1}_{ij} has to be large. Am I making any sense? Another way of presenting this might be to consider perturbations δf_j to f_j with all of them bounded by the same constant ϵ. Then the net effect on x_i is given by $\sum_{j=1}^{n} A^{-1}_{ij} \delta f_j$. Then the perturbation due to j is only significant if A^{-1}_{ij} is large in magnitude relative to the other coefficients. I may be saying the same thing you are but in a very different way. Or maybe I'm missing something here. I was not aware the AMG guys have a similar analysis but it would be helpful if you gave me a reference]

From this representation it is clear that a value, f_j, will have a strong effect on x_i if the magnitude of $(A^{-1})_{ij}$ or an appropriate measure that takes into account the magnitude of $(A^{-1})_{ij}$ is large. Note that for a given i, the coefficients, $(A^{-1})_{ij}$, $1 \leq j \leq n$, can be determined by solving the linear system,

$$A^t \mathbf{u} = \mathbf{e}_i, \tag{3.1}$$

where $\mathbf{e}_i$ is the i-th canonical vector. Then, $\mathbf{u}$ is the i-th row of A^{-1} with $u_j = (A^{-1})_{ij}$, $1 \leq j \leq n$. Note that in practice the above system is not solved exactly but instead local approximations are used. The question naturally arises whether or not, for a given i, directly using the magnitudes of the $(A^{-1})_{ij}$'s or approximations thereof is sufficient as a measure of strength of connection? It appears from numerical experiments that this is indeed sufficient (with appropriate scaling), for symmetric positive definite systems, but it tends to fail for non-symmetric systems. We do not report on those negative results in this paper, but instead present analysis and numerical experiments for a strength of connection measure based on sensitivity analysis that is suitable for both symmetric and non-symmetric systems. [**TODO**: Bobby, I have removed the term "strong connection" from the previous pages of the paper as we are, as you know, looking to identify strong couplings that later form our blocks. However, I have kept the term "strong connection" in this paragraph as it seems to be more appropriate given the

exposition before the use of the term. What do you think?(works for me) Also, I'm unclear as to the meaning of the last 2 sentences. The next to last sentence seems to say that our work doesn't work for nonsymmetric systemsI guess it came across wrong. I am referring to the measure by Brannick etal and the experiments you ran, however I am reluctant to call it out explicitly. In the symmetric case our measure defaults to theirs and both work. But in the nonsymmetric case ours is more robust and continues to work while theirs begins to break down. I hope it is clearer now, if not, let me know. We will see that it does for PageRank. However, the last sentence seems to sound like we think it will work for such systems. I'm somehow missing the intent of these two sentences. Can you help reword them?]

3.1. Motivating sensitivity analysis globally. To do this, first consider a related problem: find the effect on u_i of perturbations in elements of A. That is, we are interested in the rate of change in u_i for a change in an element of A. Here, u_i refers to the i-th entry of the solution vector, $\mathbf{u}$, of (3.1). On a global scale, one could computationally estimate such a derivative by perturbing an element in A and calculate the resulting solution uncovering the change in u_i for each such perturbation. This brute-force method demands solving such a linear system for all perturbations in A. Instead, an additional problem, the adjoint problem, is introduced which produces the desired derivatives with marked computational savings.

Given a linear system,

$$B\mathbf{y} = \mathbf{g}, \tag{3.2}$$

where B is nonsingular, let p denote a scalar parameter for which we are interested in finding $\partial y_i/\partial p$; p could be an entry, B_{ij}, of B, or g_i for that matter (although this choice will not be utilized). Differentiating the linear system with respect to p produces:

$$B\frac{\partial \mathbf{y}}{\partial p} = \frac{\partial \mathbf{g}}{\partial p} - \frac{\partial B}{\partial p}\mathbf{y}. \tag{3.3}$$

To solve for $\partial \mathbf{y}/\partial p$, consider the associated adjoint problem, $B^T\mathbf{v} = \mathbf{c}$, which easily is rewritten as

$$\mathbf{c}^T = \mathbf{v}^T B. \tag{3.4}$$

Note, for symmetric systems the adjoint system becomes $B\mathbf{v} = \mathbf{c}$, where $\mathbf{c}$ is currently unspecified. We are interested in changes to a cost functional $J(\mathbf{y}) = \mathbf{c}^T \cdot \mathbf{u}$. Differentiation gives

$$\begin{aligned} \frac{\partial J(\mathbf{y})}{\partial p} &= \mathbf{c}^T\frac{\partial \mathbf{y}}{\partial p} + \frac{\partial \mathbf{c}^T}{\partial p}\mathbf{y} \\ &= \mathbf{v}^T B\frac{\partial \mathbf{y}}{\partial p} + \frac{\partial \mathbf{c}^T}{\partial p}\mathbf{y} \\ &= \mathbf{v}^T\left(\frac{\partial \mathbf{g}}{\partial p} - \frac{\partial B}{\partial p}\mathbf{y}\right) + \frac{\partial \mathbf{c}^T}{\partial p}\mathbf{y} \end{aligned} \tag{3.5}$$

Therefore, finding $\partial J(\mathbf{y})/\partial p$ involves solving the adjoint equation (3.4) to find $\mathbf{v}$ and the linear system (3.2) to find $\mathbf{y}$. We now specialize this in the context of our application of these ideas. Choosing $B = A^T$ and $\mathbf{g} = \mathbf{e}_i$ in (3.2) implies $\mathbf{y} = \mathbf{u}$, the solution vector of (3.1). We must now decide 1) what to use as an appropriate $J(\mathbf{u})$ and p, and 2) how to localize the method.

To determine strong couplings with variable i, we are interested in the change in u_i given a change in A^T. Therefore, we take $\mathbf{c} = \mathbf{e}_i$ where $\mathbf{e}_i$ is the canonical basis vector. As such, $J(\mathbf{u}) = u_i$. Next, we will take $p = A_{jj}$, which will find the change in u_i if the value of A_{jj} were perturbed. Another choice for p is A_{ij}. Such a choice was explored and found to be a less robust indicator of strength in couplings.

With such choices determined, we can simplify (3.5). In particular, $\partial \mathbf{g}/\partial p$ and $\partial \mathbf{c}^T/\partial p$ equal $\mathbf{0}$. So, with substitutions

$$\begin{aligned} \frac{\partial u_i}{\partial A_{jj}} &= -\mathbf{v}^T\left(\frac{\partial A^T}{\partial a_{jj}}\mathbf{u}\right) \\ &= -v_j u_j. \end{aligned} \tag{3.6}$$

Now, $v_j = (A^{-1})_{ji}$ and $u_j = (A^{-1})_{ij}$, therefore

$$\frac{\partial u_i}{\partial A_{jj}} = -(A^{-1})_{ij}(A^{-1})_{ji}. \tag{3.7}$$

Note that this is a symmetric measure of strength between nodes i and j. For symmetric systems, $(A^{-1})_{ij} = (A^{-1})_{ji}$, and the measure is based only on $(A^{-1})_{ij}$. The connection of this measure to existing measures for symmetric systems is explained further in the next subsection. In practice, a normalized sensitivity is used, which is:

$$\begin{aligned} S^i_j &= \frac{p}{J(\mathbf{u})}\frac{\partial J(\mathbf{u})}{\partial p} \\ &= \frac{A_{jj}}{u_i}\left(-v_j u_j\right). \end{aligned} \tag{3.8}$$

Note that S^i_j scales the measure to account for variations in scale of A_{jj} and u_i over the domain. We have found this to be important as the unscaled measure tends to be oversensitive to such effects. Notice that after solving (3.1) and (3.4), finding $\partial u_i/\partial A_{jj}$ for all j is a trivial computation.

3.2. Theory for symmetric M-matrices. To gain some insight into the results obtained from sensitivity analysis we analyze the specific case where A is a symmetric diagonally dominant M-matrix. A is defined to be an M-matrix [30] if:

- $A_{ij} \leq 0 \ \forall i \neq j$,
- A is nonsingular,
- $A^{-1} \geq O$ (all entries of A^{-1} are non-negative).

In addition, we assume that A has constant diagonal entries, a case that for example arises when discretizing constant coefficient elliptic problems on uniform grids. Since A is symmetric, $\mathbf{u} = \mathbf{v}$, in the previous subsection. Then,

$$\begin{aligned} S^i_j &= -\frac{A_{jj}}{u_i}u_j^2 \\ &= -\frac{A_{jj}}{(A^{-1})_{ii}}((A^{-1})_{ij})^2. \end{aligned}$$

The heuristic we use for determining whether two variables are strongly coupled is given by:

$$\frac{|S^i_j|}{|S^i_{max}|} \geq \alpha, \tag{3.9}$$

where $|S^i_{max}| = \max_j |S^i_j|$ and α is a user given or adaptively varied threshold, $\alpha \in (0, 1]$. Let us first determine $|S^i_{max}|$.

$$\begin{aligned} |S^i_{max}| &= \max_j \frac{|A_{jj}|}{(A^{-1})_{ii}}((A^{-1})_{ij})^2 \\ &= \frac{|A_{ii}|}{(A^{-1})_{ii}} \max_j((A^{-1})_{ij})^2, \end{aligned}$$

since we are assuming A is an M-matrix with constant diagonal coefficients, and hence, $A_{jj} = A_{ii}$. Furthermore, since A is a diagonally dominant M-matrix, by Proposition 4.6 of [21],

$$\max_j (A^{-1})_{ij} = (A^{-1})_{ii}.$$

Hence, in this case the maximum occurs along the diagonal, and

$$|S^i_{max}| = |S^i_i| = |A_{ii}|(A^{-1})_{ii} \ \forall i.$$

We note that our criteria is almost identical to the coarsening criteria used in [6] for this very specific case. This can be seen from noting that

$$\begin{aligned} |S^i_j| &= \frac{A_{jj}}{(A^{-1})_{ii}}((A^{-1})_{ij})^2 \\ &= \frac{\|(A^{-1})_{ij}\mathbf{e}_j\|^2_A}{\|(A^{-1})_i\|^2_A}, \end{aligned}$$

and comparing with the strength of connection criterion in Theorem 1 of [6]. Here $\|\cdot\|_A$ denotes the energy norm with $\|\mathbf{w}\|^2_A = (A\mathbf{w}, \mathbf{w})$ and $(\cdot,\cdot)$ is the Euclidean inner product. Our criteria can now be further simplified to:

$$((A^{-1})_{ij})^2 \geq \alpha((A^{-1})_{ii})^2$$

which sets a lower bound on the values of $(A^{-1})_{ij}$ for variable j to be considered strongly coupled to variable i. For a particular solution variable, x_i, let us set $S = \{1, 2, \cdots, n\}$ and define subsets S_1 and S_2 of S, such that

$$S_1 = \{j : \sqrt{\alpha}(A^{-1})_{ii} \leq (A^{-1})_{ij} \leq (A^{-1})_{ii}\},$$

and $S_2 = S \setminus S_1$. Then, the componentwise solution, x_i, of (2.1) can be written as

$$x_i = \sum_{j\in S_1}(A^{-1})_{ij}f_j + \sum_{j\in S_2}(A^{-1})_{ij}f_j.$$

Since A is a M-matrix, the coefficients A^{-1}_{ij} are all non-negative. It now becomes clear from the representation above that the solution at x_i depends strongly on the values of f_j through the coefficients $(A^{-1})_{ij}$ when $j \in S_1$ and weakly when $j \in S_2$. Thus, for a symmetric M-matrix, we expect the measure of this paper to determine strong couplings to align with the heuristic followed in [6]. [**TODO**: Bobby, does this work? I wanted to be sure that the point of this analysis is clear so I added this sentence. Note again that I believe this assumes that we have a smooth f as detailed earlier.Tim, I intended the analysis is to show atleast in the M matrix case why the measure makes sense. My point was not really to show equivalence with their measure. In fact, this analysis explains why their measure makes sense! So I guess I would not really add that last sentence unless you really think it belongs there.]

3.3. Diagonal Scaling. Traditional AMG methods have trouble with diagonal scaling. In this subsection we show that our measure is invariant with respect to diagonal scaling of the matrix A. The proof is simple and follows along similar lines to that outlined in Theorem 1 of [6]. Note that we do not require A to be symmetric.

LEMMA 3.1. *Let D be a diagonal $n \times n$ matrix and $\tilde{A} = DAD$. Let $\tilde{A}\tilde{\mathbf{u}} = \mathbf{e}_i$ and $\tilde{A}^t\tilde{\mathbf{v}} = \mathbf{e}_i$. Define the sensitivity measures $\tilde{S}^i_j$ and S^i_j for $\tilde{A}$ and A. Then, $\tilde{S}^i_j = S^i_j$.*

Proof. It is easy to see that $\tilde{u}_i = D^{-2}_{ii}u_i$, $\tilde{u}_j = D^{-1}_{jj}D^{-1}_{ii}u_j$, and $\tilde{v}_j = D^{-1}_{jj}D^{-1}_{ii}v_j$. Then,

$$\begin{aligned} \tilde{S}^i_j &= -\frac{\tilde{A}_{jj}}{\tilde{u}_i}\tilde{v}_j\tilde{u}_j \\ &= -\frac{D^2_{jj}A_{jj}}{D^{-2}_{ii}u_i}(D^{-1}_{jj}D^{-1}_{ii}v_j)(D^{-1}_{jj}D^{-1}_{ii}u_j) \\ &= -\frac{A_{jj}}{u_i}v_ju_j \\ &= S^i_j. \end{aligned}$$

□

3.4. Local Sensitivity Analysis (LSA). Clearly, localization of such a method is necessary for computational viability. The goal is to create a sufficiently efficient algorithm while uncovering the desired information needed to determine strong couplings.

We define the distance 1-neighborhood of i to be the index set $C_i = \{i\} \cup \{j \mid A_{ij} \neq 0\}$ with cardinality $m \leq n$. Corresponding to this 1-neighborhood we assemble a local $m \times m$ submatrix A_i formed by removing from A all rows j and columns j if j does not belong to C_i. More formally, form the $n \times m$ matrix,

$$P_i = [\mathbf{e}_{j_1} \mathbf{e}_{j_2} \cdots \mathbf{e}_{j_m}],$$

where $\mathbf{e}_{j_k}, k = 1, 2, \cdots m, j_k \in C_i$, are the canonical column basis vectors with all zeros except for a 1 in location j_k. Then

$$A_i \equiv P_i^t A P_i.$$

Sensitivity analysis is performed on the local linear system,

$$A_i^t \mathbf{u} = \mathbf{e}_i, \tag{3.10}$$

where it is assumed that A_i is non-singular. [**TODO**: Bobby, should this either be A_i^T or $\mathbf{v}$ to correspond to the notation in (3.1) and (3.4)?Tim, notice the P^t has disappeared. Does that look right to you? Thanks for catching this one]

We refer to this local process as Local Sensitivity Analysis (LSA). We make note of three important points.

- It is clearly possible to further generalize by considering distance 2− or distance 3− neighborhoods of i but this adds computational cost and was found to be unnecessary in our experiments. The information obtained by LSA on 1− neighborhoods was found to be sufficient for our purposes.
- For constant coefficient problems it is obviously sufficient to perform LSA on a small set of local linear systems (one interior system and one local system for each boundary) making the cost negligible. For variable coefficient systems LSA needs to be performed for all variables. In such cases the cost of LSA is similar to what a block Jacobi smoothing operation might incur.
- LSA is an embarrassingly parallel operation as it is involves solving multiple decoupled local systems, which involves no interprocessor communication.

Using LSA in the local manner described we show how we can uncover strong couplings in a wide variety of problems. We now focus on using such knowledge to create block smoothers algebraically. The next section motivates and develops such a method.

3.5. Two model examples. While the ideas of the previous section can be applied to scalar and system PDEs and also nonsymmetric problems, we focus on two classical problems in multigrid analysis. This section motivates the use of such ideas for the block smoothing algorithm to follow.

3.5.1. Isotropic Laplacian. Consider the Laplacian equation

$$u_{xx} + \epsilon u_{yy} = 0 \tag{3.11}$$

on a square grid, discretized with a standard 9-point stencil. If variable i be in the interior of the domain, then

$$A_i = \begin{pmatrix} 1 & -\frac{1}{8} & 0 & -\frac{1}{8} & -\frac{1}{8} & 0 & 0 & 0 & 0 \\ -\frac{1}{8} & 1 & -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & 0 & 0 & 0 \\ 0 & -\frac{1}{8} & 1 & 0 & -\frac{1}{8} & -\frac{1}{8} & 0 & 0 & 0 \\ -\frac{1}{8} & -\frac{1}{8} & 0 & 1 & -\frac{1}{8} & 0 & -\frac{1}{8} & -\frac{1}{8} & 0 \\ -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & 1 & -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} \\ 0 & -\frac{1}{8} & -\frac{1}{8} & 0 & -\frac{1}{8} & 1 & 0 & -\frac{1}{8} & -\frac{1}{8} \\ 0 & 0 & 0 & -\frac{1}{8} & -\frac{1}{8} & 0 & 1 & -\frac{1}{8} & 0 \\ 0 & 0 & 0 & -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{8} & 1 & -\frac{1}{8} \\ 0 & 0 & 0 & 0 & -\frac{1}{8} & -\frac{1}{8} & 0 & -\frac{1}{8} & 1 \end{pmatrix},$$

which also results in variable i have 9 distance 1-neighbors. As discussed in Section 3.2, the largest S_j^i will occur when $j = i$. Again, we scale these values such that the largest $S_j^i = 1$. We report these values in the stencil form below:

$$\begin{pmatrix} 0.033 & 0.052 & 0.033 \\ 0.052 & 1.0 & 0.052 \\ 0.033 & 0.052 & 0.033 \end{pmatrix}$$

as a compact notation for reading S_j^i for any j in the 1-neighborhood of i. Explicitly the stencil encodes $S_j^i = 0.052$ for the variables j that are to the north, south, east and west of variable i. Similarly, the variables to the northwest, northeast, southeast and southwest of i produce $S_j^i = 0.033$. Such results would be expected by any algorithm proposing to yield information regarding strength for this problem.

3.5.2. Anisotropic Laplacian. Now, let us set $\epsilon = 1/100$ in (3.11). Now, if variable i is in the interior of the domain, then

$$A_i = \begin{pmatrix} 8.0 & 1.9 & 0 & -3.9 & -1 & 0 & 0 & 0 & 0 \\ 1.9 & 8.0 & 1.9 & -1 & -3.9 & -1 & 0 & 0 & 0 \\ 0 & 1.9 & 8.0 & 0 & -1 & -3.9 & 0 & 0 & 0 \\ -3.9 & -1 & 0 & 8.0 & 1.9 & 0 & -3.9 & -1 & 0 \\ -1 & -3.9 & -1 & 1.9 & 8.0 & 1.9 & -1 & -3.9 & -1 \\ 0 & -1 & -3.9 & 0 & 1.9 & 8.0 & 0 & -1 & -3.9 \\ 0 & 0 & 0 & -3.9 & -1 & 0 & 8.0 & 1.9 & 0 \\ 0 & 0 & 0 & -1 & -3.9 & -1 & 1.9 & 8.0 & 1.9 \\ 0 & 0 & 0 & 0 & -1 & -3.9 & 0 & 1.9 & 8.0 \end{pmatrix}.$$

Again we report, in stencil form, S_j^i for all j in the neighborhood of i where the values are scaled such that the largest $S_j^i = 1$:

$$\begin{pmatrix} 0.010 & 0.237 & 0.010 \\ 0.049 & 1.0 & 0.049 \\ 0.010 & 0.237 & 0.010 \end{pmatrix}$$

Again, the strongest coupling to the variable i is itself. However, this problem demonstrates much stronger coupling to the north and south. Much weaker connections exist to the east, west and diagonal connections, which is expected for this problem. Note, the original stencil is of the type that has positive and negative off-diagonals, which can pose more troublesome to AMG's traditional measure of strength. [**TODO**: Bobby, can you check this as I have changed to the PDE rather than talking about discretizing with square and rectangles. Does this work?]

More examples will be considered later in this paper in the context of using such a measure of strength to produce an adaptive algebraic smoother for a variety of applications that include system PDEs and nonsymmetric problems.

3.6. Algorithm for identifying strongly coupled blocks. The pseudo code contained in Algorithm 1 presents the steps followed to determine the blocks for smoothing for the section to follow. For the example of the anisotropic Laplacian using this algorithm we are able to form blocks which correspond to the lines which a line smoother would use when $\alpha < 0.237$ (see section on numerical experiments). Heuristically we have determined that setting $\alpha = 0.1$ provides good results for most cases. In a fully adaptive algorithm (as we will describe in a related publication) α can be varied to obtain a whole class of smoothers for a given problem.

```
Initialize:
n ⟵ 1                                          /* n:  indexes the number of blocks */
B ⟵ ∅                                          /* B:  the set of blocks */
B_n ⟵ ∅                                        /* B_n:  n-th block being formed */
B_new ⟵ {1}                                    /* B_new:  set of new indices to add to block B_n */
F ⟵ ∅                                          /* F:  set of strong connections to indices in B_n */
S ⟵ {2, ··· , n}
Iterate:
while S! = ∅ do                                /* While S contains unprocessed indices */
    for j ∈ B_new do
        for k ∈ (C_j \ {k}) ∩ S do             /* 1-neighborhood of j */
            if S_k^j ≥ α then                  /* if strongly connected */
                F ⟵ F ∪ {k}
                S ⟵ S \ {k}
            end
        end
    end
    B_n ⟵ B_n ∪ B_new                          /* Add new indices to B_n */
    if F! = ∅ then
        B_new ⟵ F
        F ⟵ ∅
    else
        B ⟵ B ∪ {B_n}
        n ⟵ n + 1
        B_n ⟵ ∅
        F ⟵ ∅
        B_new ⟵ {head(S)}                      /* head(S) refers to next element in S */
    end
end
```

Algorithm 1: Determining Blocks for Smoother

3.7. Forming an Adaptive Algebraic Smoother. The goal of this section is to utilize the information available from LSA to form a smoother. The algebraic nature of our methods means we are concerned with an algebraic sense of smoothness and not geometric smoothness. However, we will see that in some cases the algebraic and geometric sense of smoothness do coincide. In particular, for the purposes of smoothing, strongly connected variables will be grouped together to form blocks. These blocks will then be used to define a stationary block iterative scheme. We assume that we have been able to identify groups of strongly connected variables using sensitivity analysis . Grouping these variables together into a block is essentially a reordering of the matrix A. Let P denote a $n \times n$ permutation matrix that reorders A so that blocks of strongly coupled variables are grouped together. Then the reordered matrix is given by $P^t AP$. For simplicity of notation, in the rest of this section we

will use A to denote the reordered matrix P^tAP also. Now, let

$$A = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1s} \\ A_{21} & A_{22} & \cdots & A_{2s} \\ \cdots & \cdots & \cdots & \cdots \\ A_{s1} & \cdots & A_{s,s-1} & A_{ss} \end{bmatrix}$$

where A_{ij}, $1 \leq i, j \leq s$ are now s^2 matrix subblocks of size $q_i \times q_j$. Then, $\sum_{i=1}^{s} q_i = n$. A block iterative stationary method can now be defined by a splitting, $A = M - N$, where M is invertible, which leads to the stationary iteration:

$$\mathbf{x}^{k+1} = M^{-1}N\mathbf{x}^k + M^{-1}\mathbf{f}.$$

The vectors $\mathbf{x}^{k+1}$, $\mathbf{x}^k$, and $\mathbf{f}$ are partitioned to match the block partitioning of the matrix A. The iteration given above converges if and only if $\rho(M^{-1}N) < 1$.

An example of a block iteration is the block Gauss-Seidel method defined by the splitting:

$$M = \begin{bmatrix} A_{11} & 0 & \cdots & 0 \\ A_{21} & A_{22} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ A_{s1} & \cdots & A_{s,s-1} & A_{ss} \end{bmatrix}$$

and $N = M - A$. When $s = n$, then the subblocks are of size one and the block iterations reduces to the standard lexicographic Gauss-Seidel iteration. The numerical results presented in this paper were all performed with block Gauss-Seidel iterations, though once the block partitioning is defined we are free to choose any suitable block iterative process.

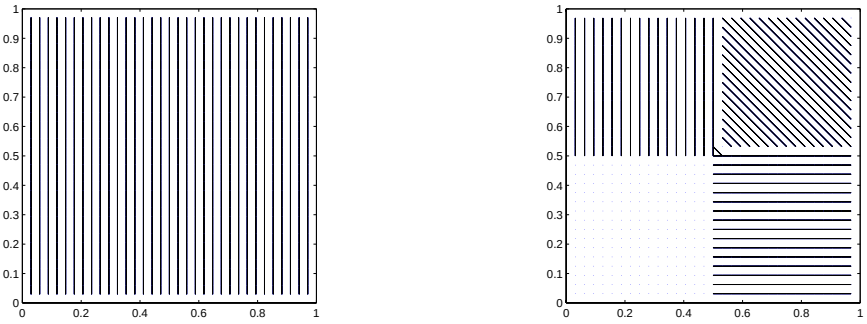
However, an important subtlety arises: we must be able to identify variables for which block smoothing is not necessary. Heuristically, such variables are sufficiently "strongly connected" to themselves; the isotropic Poisson operator is a motivating example for such problems. To illustrate this we return to the examples in the previous section. For $\alpha > 0.237$ we see that both the anisotropic and isotropic systems view only variable i as being strongly connected to itself. In this case we set the block size to be 1, i.e. no block smoothing is required. We note that this would be a perfectly acceptable choice for the anisotropic system if semi-coarsening is used within a multigrid algorithm [27]. On the other hand, if α lies in $(0.052, 0.237)$, then the isotropic system will still indicate that variable i is strongly connected only to itself, while the anisotropic system will indicate strongly connected variables to the north and south of i.

4. Numerical Experiments. This section reports numerical results for the algebraic block smoothing algorithm introduced in Section 3.7. For each experiment, this new method's performance will be compared to that of pointwise Gauss-Seidel.

Unless otherwise stated $\alpha = 0.1$. Unless otherwise stated, we solve the homogeneous problem ($\mathbf{f} = \mathbf{0}$) with a random initial vector, $\mathbf{x}^0$, with entries between 0 and 1. A convergence factor which is the ratio $\left(\|\mathbf{r}^k\|_2/\|\mathbf{r}^{k-1}\|_2\right)$ for each iteration k is also included.

Keep in mind that such methods can be used to complement a multigrid method that dampens remaining components of the error with a coarse-grid correction process. This paper focuses on such methods as linear solvers of themselves and in some cases, we will see that the methods succeed as such a solver.

In the numerics to follow, we look at three main classes of problems: scalar PDEs, coupled system PDEs and linear systems related to the PageRank vector used by search engines such as Google. Within these classes of problems, we will consider nonsymmetric systems. Note that system PDEs and nonsymmetric linear systems are traditionally difficult for multigrid methods to solve efficiently. It is notable that the smoothers of this paper can be extended to such problems without adjustment to the algorithm.



(a) (b)

FIG. 4.1. *Lines created by adaptive algebraic smoother for (a) anisotropic Poisson problem and (b) PDE defined in (4.1) with anisotropies varying in four regions of the domain.*

4.1. Scalar Partial Differential Equations. To begin, we demonstrate the ability of such a method to produce blocks desirable for scalar PDEs.

4.1.1. Isotropic Laplacian. We begin by considering the isotropic Laplacian again from (3.11) with $\epsilon = 1$. As stated in Section 3.5.1, S_j^i for all j in the neighborhood of i where the values are scaled such that the largest $S_j^i = 1$, reported in stencil form is:

$$\begin{pmatrix} 0.033 & 0.052 & 0.033 \\ 0.052 & 1.0 & 0.052 \\ 0.033 & 0.052 & 0.033 \end{pmatrix}$$

With $\alpha = 0.1$, each point is considered strongly coupled only to itself. As such, the method produces a pointwise Gauss-Seidel method, which would be expected for this problem.

4.1.2. Anisotropic Laplacian. We now consider system (3.11) for $\epsilon = 1/100$. As earlier in the paper, reporting, in stencil form, S_j^i for all j in the neighborhood of i where the values are scaled such that the largest $S_j^i = 1$:

$$\begin{pmatrix} 0.010 & 0.237 & 0.010 \\ 0.049 & 1.0 & 0.049 \\ 0.010 & 0.237 & 0.010 \end{pmatrix}$$

While the strongest coupling to the variable i is itself, a strong connection also exists to the north and south. Finding the strength of couplings in this way for both interior and boundary variables, creates the lines seen in Figure 4.1 (a). As such, the method creates the line smoother considered a standard option for such a problem. We see in Table 4.1 how the method compares to pointwise Gauss-Seidel. For such a problem, a multigrid method would overcome the slower convergence of pointwise Gauss-Seidel with semi-coarsening.

4.1.3. Varying anisotropy within the domain. We next turn to an example from Section 1.3 in [25]. The underlying PDE is:

$$-(au_x)_x - (bu_y)_y + cu_{xy} = f(x, y) \tag{4.1}$$

defined on a unit square with full Dirichlet boundary conditions. The problem is defined such that $a = b = 1$ everywhere except in the upper left quarter of the unit square where $b = 10^3$ and the lower right quarter where $a = 10^3$. To split the domain into four regions with varying anisotropies, $c = 0$ except in the upper right quarter where $c = 2$.

Pointwise		Block	
$\|r\|$	Conv. Fac.	$\|r\|$	Conv. Fac.
172.42880	1.00000	172.42880	1.00000
37.17199	0.21558	9.01827	0.05230
11.27923	0.30343	0.60601	0.06720
5.05070	0.44779	0.05297	0.08740
3.21387	0.63632	0.00659	0.12440

TABLE 4.1

Equation (3.11) on a 35×35 *rectangular grid with full Dirichlet boundary conditions, discretized with bilinear quadrilateral elements, with* $\epsilon = 1/00$. *Algebraic block smoothing created 33 blocks.*

Pointwise		Block	
$\|r\|$	Conv. Fac.	$\|r\|$	Conv. Fac.
1.704e+007	1.00000	1.704e+007	1.00000
2.580e+006	0.15138	2.908e+003	0.00017
7.049e+005	0.27318	9.142e+002	0.31444
3.736e+005	0.53000	6.060e+002	0.66285
2.666e+005	0.71357	4.764e+002	0.78607

TABLE 4.2

Pointwise and block smoothing results for PDE given in 4.1 defined with full Dirichlet boundary conditions. Algebraic block smoothing created 285 blocks.

The discretized system is formed using a standard 5-point stencil and a (left-oriented) 7-point stencil for the diffusion and mixed derivative points of the PDE, respectively. As a result of these varying coefficients, the system is isotropic in the lower left quarter of the unit square but strongly anisotropic in the remaining quarters. The direction, however, of the anisotropy varies in the remaining three quarters of the unit square with the direction of strong connection lying in the x, y and diagonal directions for the upper left, lower right, and upper right quarters, respectively. The varying directions of these anisotropies are reflected in the smooth error produced after four iterations of pointwise Gauss-Seidel seen in Figure 4.2 (a). Note that the numerics use the discretized system which included both A and $\mathbf{f}$ were supplied by Klaus Stüben.

In Table 4.2, pointwise Gauss-Seidel converges toward the solution. However, the adaptive algebraic smoother performs much better particularly in the first iteration which is important for multigrid methods. The adaptive algebraic smoother forms blocks that geometrically follow the anisotropies within each region as seen in Figure 4.1 (b), which results in geometrically smooth error, as seen in Figure 4.2 (b), suggesting its usefulness for geometric multigrid methods. Note that the largest error occurs in the quarter of the domain that is isotropic which is where the block smoother chooses only pointwise smoothing.

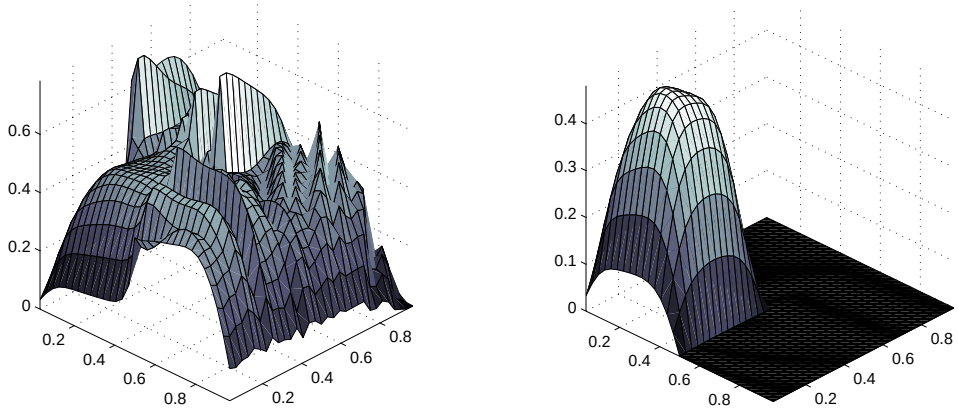
4.1.4. Discontinuous coefficients. We next turn to an example with strongly discontinuous coefficients. The following model problem comes from Section 8.4.1 in [25]. The underlying diffusion problem is:

$$-(au_x)_x - (bu_y)_y = f(x, y), \tag{4.2}$$

on a unit square with discontinuous coefficients $a > 0$ and $b > 0$ as defined in Figure 4.3. Note that $f(x, y) = 0$ except at the points $(0.25, 0.25), (0.5, 0.5)$ and $(0.75, 0.75)$ where $f(x, y) = 10$. The problem has the following Dirichlet boundary conditions

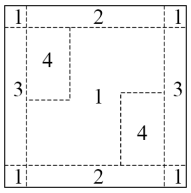
$$u = 1 \quad \text{for} \quad x \leq 0.5,\ y = 0 \text{ and } x = 0,\ y \leq 0.5; \quad \text{otherwise } u = 0.$$

The system is discretized using a standard 5-point stencil on a regular grid with a mesh of $h = 1/N$. The numerics use the discretized system supplied by Klaus Stüben.



(a) (b)

FIG. 4.2. *Algebraically smooth error after four iterations of (a) pointwise smoother and (b) block smoother for the scalar PDE (4.1).*



region 1: $a = 10^{-3}$, $b = 10^{-3}$
region 2: $a = 10^{+3}$, $b = 10^{-3}$
region 3: $a = 10^{-3}$, $b = 10^{+3}$
region 4: $a = 1$, $b = 1$

FIG. 4.3. *Distribution of discontinuous coefficients in (4.2).*

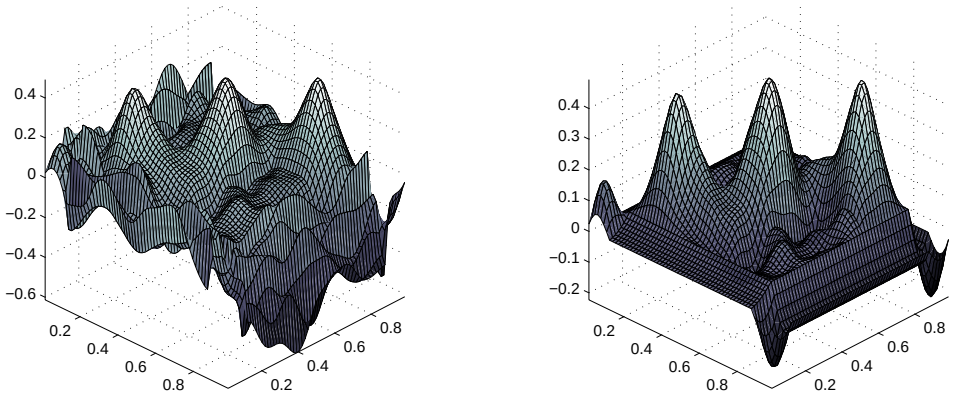
In Table 4.3, pointwise Gauss-Seidel converges toward the solution. However, the adaptive algebraic smoother performs much better particularly in the first iteration which is important for multigrid methods. Further, the block smoother produces geometrically smooth error as seen in Figure 4.4 suggesting its usefulness for geometric multigrid methods. However, it should be noted that 2, 28 and 1 blocks contain 47, 49 and 192 variables, respectively with the remaining variables undergoing pointwise smoothing.

The results of this linear system foreshadow the potential need to alter α. These tests involved solving a linear system for each block, and the numerical results of this paper utilized a direct solve. As such, if the size of the largest block for $\alpha = 0.1$ is undesirable, several options exist to reduce the computational cost of the block smoother. First, such large blocks could be attacked as a smaller linear systems and solved iteratively. Alternatively, the adaptive algebraic smoother could break large blocks into overlapping blocks where the amount of overlap can result in better convergence [5]. Another option is to alter α in (3.9). While here we will increase α, an algorithm could start with a relatively large α and adaptively reduce this size of this threshold until either blocks become unsuitably large or a desired level of convergence results. Here, again, we will raise $\alpha = 0.24995$. This value of α results in 2,533 blocks with 47 blocks containing 2 variables and 28 blocks containing 49 variables. Note that the block of 192 no longer remains but became a collection of smaller blocks. While the larger α results in slower convergence, especially in the second iterate, the method still performs well in comparison to pointwise Gauss-Seidel. It should be noted that $\alpha \leq 0.24994$ results in the large 192 sized block remaining and $\alpha \geq 0.25$ results in only a few blocks that contain more than one point. The next example considers another application where adaptively altering α would serve advantageous.

Pointwise		Block	
$\|r\|$	Conv. Fac.	$\|r\|$	Conv. Fac.
1.14838e+008	1.00000	1.14838e+008	1.00000
1.69170e+007	0.14731	1.71013e+004	0.00015
3.86860e+006	0.22868	3.78435e+003	0.22129
1.65386e+006	0.42751	1.72486e+003	0.45579
1.01471e+006	0.61354	1.07186e+003	0.62142

TABLE 4.3

Pointwise and block smoothing results for PDE given in (4.2) defined with strongly discontinuous coefficents. Algebraic block smoothing created 2342 blocks with $\alpha = 0.1$.



(a) (b)

FIG. 4.4. *Algebraically smooth error after four iterations of (a) pointwise smoother and (b) block smoother for the scalar PDE (4.2) with strongly discontinuous coefficients with* $\alpha = 0.1$ *for the adaptive block smoother.*

4.1.5. Oil reservoir modeling system. The following linear system (consisting of both A and f) is from the Harwell-Boeing collection available from Matrix Market (`http://math.nist.gov/MatrixMarket/`). It is the SHERMAN2 matrix which comes from a three dimensional thermal simulation with steam injection (from the description on the website) and is one of the oil reservoir simulation challenge matrices from Andy Sherman. The matrix A is a 1080×1080 real non-symmetric matrix with 23094 non-zero entries. It is not diagonally dominant and has a condition number estimate of $1.4e + 12$. Gauss-Seidel diverges for this problem due to the lack of diagonal dominance. Table 4.5 shows the convergence history as we vary the threshold, α, from 0.2 to 0.05. For $\alpha = 0.2$ the block iteration does not converge at all, while for $\alpha = 0.1$ we obtain marginally good convergence of the smoother. By then lowering α even further to 0.2 we are able to obtain good convergence for this system. Table 4.6 shows that as α is varied the number and size of the blocks can vary considerably. This example is not meant to imply that larger block sizes necessarily lead to better convergence rates. Counter examples are provided by Varga[30] where this is not the case. However, larger block sizes *can* lead to better convergence rates as this experiment demonstrates. Furthermore, this example illustrates how α might be reduced by an adaptive algorithm, which should balance convergence and work requirements in the resulting block iterative smoother. Research in multigrid methods that construct multigrid components based on measures that strive to balance convergence and work include [9] and [20].

Pointwise		Block	
$\|r\|$	Conv. Fac.	$\|r\|$	Conv. Fac.
1.14838e+008	1.00000	1.14838e+008	1.00000
1.69170e+007	0.14731	4.31595e+006	0.03758
3.86860e+006	0.22868	9.89439e+005	0.22925
1.65386e+006	0.42751	4.55776e+005	0.46064
1.01471e+006	0.61354	3.10844e+005	0.68201

TABLE 4.4

Pointwise and block smoothing results for PDE given in (4.2) defined with strongly discontinuous coefficents. Algebraic block smoothing created 2342 blocks with $\alpha = 0.24995$.

$\alpha = 0.2$		$\alpha = 0.1$		$\alpha = 0.05$	
$\|r\|$	Conv. Fac.	$\|r\|$	Conv. Fac.	$\|r\|$	Conv. Fac.
1.74403e+08	1.00000e+00	1.74403e+08	1.00000e+00	1.74403e+08	1.00000e+00
1.74810e+06	1.00233e-02	1.74239e+06	9.99058e-03	1.31409e+05	7.53480e-04
1.66690e+05	9.53548e-02	1.69416e+05	9.72317e-02	3.63905e+02	2.76924e-03
6.23284e+04	3.73918e-01	1.68578e+04	9.95054e-02	2.98207e+00	8.19464e-03
1.10653e+06	1.77532e+01	1.71429e+03	1.01691e-01	1.67902e-01	5.63039e-02

TABLE 4.5

Convergence of block smoothers for SHERMAN2 problem with varying α.

4.2. Systems of Partial Differential Equations. System PDEs are traditionally difficult problems for algebraic multigrid methods. This section demonstrates the ability of the adaptive algebraic smoother to define strong couplings for such problems and its affect on the block smoothing iteration. For simplicity the examples presented in this section are 2×2 coupled systems of PDEs where AMG is known to have trouble.

4.2.1. Model systems. In this section, we consider two model systems where the anisotropy in each variable differs. These systems were provided by Jim E. Jones [14]. In order to introduce the system of interest, define

$$-\triangle_{\epsilon:x} \equiv \begin{pmatrix} & -1 & \\ -\epsilon & 2+2\epsilon & -\epsilon \\ & -1 & \end{pmatrix} \quad \text{and} \quad -\triangle_{\epsilon:y} \equiv \begin{pmatrix} & -\epsilon & \\ -1 & 2+2\epsilon & -1 \\ & -\epsilon & \end{pmatrix}.$$

Therefore, this section considers the linear system:

$$\begin{pmatrix} -\triangle_{\epsilon:x} & kI \\ -kI & -\triangle_{\epsilon:y} \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix}. \tag{4.3}$$

For the numerics to follow, we will set $\epsilon = 0.01$.

In the first system, the diagonal blocks have the 5-point anisotropic Laplacian with $\epsilon = 0.01$. Again, the direction of the anisotropy differs between the blocks. In order to have a coupled system, we set $k = 0.01$.

In Table 4.7, pointwise Gauss-Seidel converges toward the solution. However, the adaptive algebraic smoother performs much better both in early and later iterations. The system is 2178×2178 with a total of 66 blocks where each block contains 33 variables.

In the second system, we keep $\epsilon = 0.01$. However, we create a large off-diagonal element by setting $k = 100$. In fact, the linear system is no longer (and far from being) diagonally dominant. The adaptive algebraic smoother's success on problems difficult for pointwise Gauss-Seidel is clearly seen in this example. In Table 4.8 we see clear divergence for pointwise Gauss-Seidel. The adaptive algebraic smoother performs with low convergence rates. Again, the system is 2178×2178. The block smoother chooses 1089 blocks with each being a 2×2 block.

α	Number of blocks	Max. block size	Min. block size
0.2	292	44	1
0.1	268	45	1
0.05	241	150	1

TABLE 4.6
Block statistics for SHERMAN2 problem with varying α.

Pointwise		Block	
$\|r\|$	Conv. Fac.	$\|r\|$	Conv. Fac.
33.43388	1.00000	33.43388	1.00000
4.91278	0.14694	0.06436	0.00193
1.32823	0.27036	0.02394	0.37189
0.73113	0.55045	0.00427	0.17855
0.53874	0.73686	0.00204	0.47660

TABLE 4.7
For $\alpha = 0.1$, algebraic block smoothing created 66 blocks for $\epsilon = 0.01$ and $k = 0.01$ in system (4.3).

4.2.2. Linear elasticity. Next, we apply the adaptive algebraic block smoothing method to the 2D linear elasticity system

$$\begin{aligned} u_{xx} + \frac{1-\beta}{2} u_{yy} + \frac{1+\beta}{2} v_{xy} &= f_1, \\ \frac{1+\beta}{2} u_{xy} + \frac{1-\beta}{2} v_{xx} + v_{yy} &= f_2, \end{aligned} \tag{4.4}$$

where u and v are displacements in the x and y directions, respectively. Throughout the numerical tests, we employ the value $\beta = 1/2$, which yields the Poisson ratio $\nu = \beta/(1+\beta) = 1/3$. The problem has free boundaries, except on the left where $u = v = 0$. We discretize (4.4) with bilinear finite elements on a uniform $n_x \times n_y$ rectangular array of cells with spacing $h_x \times h_y$. The actual domain ($n_x h_x$ by $n_y h_y$) varies in size, depending on the values of n_x, n_y, h_x, and h_y.

We begin by discretizing with square elements ($h_x = 1/32$ and $h_y = 1/32$) on a 32×32 grid. We see from Table 4.9 that the adaptive algebraic smoother has efficient dampening in the early iterations which again bodes well to its role in the multigrid process. It should be noted, however, that the average block size is 17 with 29, 30 and 1 blocks of sizes 32, 33, and 194, respectively; the remaining blocks consisted of a single variable.

We next stretch the grid and consider linear elasticity discretized with stretched rectangular elements ($h_x = 1/32$ and $h_y = 1/320$) on a 32×32 grid. We see again that the method perform well in the early iterations. In this case, the method also performs well in later iterates. Note that the block smoother has 62 and 1 blocks of sizes 33 and 66, respectively with the remaining blocks consisting of a single variable.

Next, the method is tested on the single-element thick 2-D plane-stress cantilever beam discretized with square elements on a 64×1 grid. The adaptive algebraic smoother created 5 blocks with 256 of the total 260 variables residing in a single block. As such, the method served essentially as a direct solver and the block smoother converged in a single iterate. Therefore, α was raised to 0.3 which produces the results in Table 4.11. In this case more (and much smaller) blocks are created. Note that the linear system is 256×256 so many of the blocks consist of only one variable. More specifically, the blocks consist of 62 blocks containing 2 variables and a single 8×8 block. The remaining 128 degrees of freedom have pointwise smoothing. We see from the iterates that after 4 smoothing steps, the residual is approximately half that of the residual resulting from pointwise Gauss-Seidel.

We see similar behavior for the single-element thick 2-D plane-stress cantilever beam with stretched rectangular elements with a $10:1$ aspect ratio. For this problem, $\alpha = 0.1$ results again in a large

Pointwise		Block	
$\|r\|$	Conv. Fac.	$\|r\|$	Conv. Fac.
2.68835e+003	1.00000e+000	2.68835e+003	1.00000
2.89686e+009	1.07756e+006	2.65285e-001	0.00010
1.06165e+014	3.66484e+004	2.61072e-005	0.00010
3.94519e+018	3.71608e+004	2.56705e-009	0.00010
1.47601e+023	3.74130e+004	2.51757e-013	0.00010

TABLE 4.8

For $\alpha = 0.1$, the adaptive algebraic smoother created 1089 blocks for $\epsilon = 0.01$ and $k = 100$ in system (4.3).

Pointwise		Block	
$\|r\|$	Conv. Fac.	$\|r\|$	Conv. Fac.
98.34003	1.00000	98.34003	1.00000
15.10429	0.15359	6.68950	0.06802
4.26136	0.28213	1.39356	0.20832
1.89268	0.44415	0.62380	0.44763
1.19331	0.63049	0.43789	0.70197

TABLE 4.9

Linear elasticity discretized with square elements ($h_x = 1/32$ and $h_y = 1/32$) on a 32×32 grid. Algebraic block smoothing created 126 blocks.

block of of size 256 (of the total 260 variables). Raising α to 0.3 results in 129 blocks producing the numerics in Table 4.12. Of the 126 total blocks formed, 124 and 1 blocks contained 2 and 8 variables, respectively with the remaining 4 blocks containing a single variable. Note that pointwise Gauss-Seidel performs much better for the first iteration but then degrades. This was true over multiple runs with various random initial guesses.

4.3. Linear Systems related to PageRank. Finally, we consider linear systems associated with search engine analysis and the PageRank algorithm. At the core of the PageRank algorithm is a Markov Chain model of internet activity. The states of the Markov process are web pages. To form the transition matrix, some variables are defined. Let W be a connected network of n web pages. Let G be the $n \times n$ adjacency matrix of W, that is, g_{ij} is 1 if there is a hyperlink to page i from page j and 0 otherwise. Note, G is a sparse matrix. Let c_j and r_i be the column and row sums of G, respectively. That is,

$$c_j = \sum_{1 \le i \le n} g_{ij}, \qquad r_i = \sum_{1 \le j \le n} g_{ij}$$

Then c_k and r_k are the out-degree and in-degree of the web page corresponding to row k of G. (We will call this web page k.)

To form the transition matrix M that is fundamental to the Markov process, a probability of going from page i to page j must be defined. (Note, we will form a column stochastic matrix.) Define the matrix H where

$$h_{ij} = \begin{cases} g_{ij}/c_i & \text{if } c_i \neq 0, \\ g_{ij} & \text{if } c_i = 0. \end{cases}$$

Note that H is not a transition matrix as it contains zero columns due to the presence of *dangling nodes*, which have an out-degree of 0.

Let $\mathbf{v} = \frac{1}{n}\mathbf{1}$. Also, define $\mathbf{d}$ such that

$$d_i = \begin{cases} 1 & \text{if } c_i = 0, \\ 0 & \text{otherwise.} \end{cases}$$

Pointwise		Block	
$\|r\|$	Conv. Fac.	$\|r\|$	Conv. Fac.
616.50672	1.00000	616.50672	1.00000
138.17378	0.22412	34.59954	0.05612
42.00994	0.30404	2.54766	0.07363
16.60365	0.39523	0.27511	0.10799
8.67694	0.52259	0.10852	0.39445

TABLE 4.10

Linear elasticity discretized with stretched rectangular elements ($h_x = 1/32$ and $h_y = 1/320$) on a 32×32 grid. Algebraic block smoothing created 129 blocks.

Pointwise		Block	
$\|r\|$	Conv. Fac.	$\|r\|$	Conv. Fac.
17.88577	1.00000	17.88577	1.00000
1.89137	0.10575	2.21025	0.12358
0.87261	0.46136	0.75029	0.33946
0.55868	0.64024	0.35001	0.46650
0.41424	0.74147	0.21359	0.61025

TABLE 4.11

Single-element thick 2-D plane-stress cantilever beam discretized with square elements on a 64×1 grid. For $\alpha = 0.3$, the algebraic block smoothing created 189 blocks.

Therefore, $S = H + \mathbf{v}\mathbf{d}^T$. Now, the transition matrix $G \equiv \alpha S + (1 - \alpha)\mathbf{v}\mathbf{1}^T$, where $\alpha = 0.85$, which represents the fraction of time that a random walk through the network follows a link available on a web page. The matrix G is called the Google matrix.

Therefore, the steady state vector of the Markov process is $G\mathbf{x} = \mathbf{x}$. In Google's case, a stochastic $\mathbf{x}$ gives the PageRank with the largest element of $\mathbf{x}$, say x_i, corresponding to the web page (web page i in this case) with the highest PageRank. The second largest element of $\mathbf{x}$ corresponds to the web page with the second highest PageRank and so on.

In this paper, we look at a corresponding linear system which preserves the sparsity of G. In particular, we are interested in the system:

$$(I - \alpha H)\mathbf{y} = \mathbf{v}. \tag{4.5}$$

Note, $I - \alpha H$ is a nonsingular M-matrix. After solving this linear system, the PageRank vector $\mathbf{x}$ is then computed by letting $\mathbf{x} = \mathbf{y}/\mathbf{y}\mathbf{1}^T$. For a more thorough discussion on the properties of this matrix and for a proof that such a process produces the PageRank vector, see [17].

This paper will restrain from applying the method of this paper to a large set of sample networks downloaded from [28]. The purpose of this paper is to demonstrate the effectiveness of the method for representative samples of such networks.

Each collection of web pages was created following the guidelines of Kleinberg [16]. The search engine AltaVista was queried for one or more keywords. Note that when a query consisted of more than one word, the '+' symbol used to ensure that every page in the network contained the keywords. The first 200 pages returned by AltaVista form what is called the *Root Set*. For each page in the Root Set, the out-links of that page are stored as well as the first 50 in-links, in the order AltaVista returns them. The Root Set is expanded into the Base Set by including the in-links, and out-links of the pages in the Root Set. After forming the Base Set, the underlying graph of the network of websites is constructed. Edges that connect two nodes within the same domain were deleted since they usually serve navigation purposes. Isolated nodes were also deleted. For more information, see [29, 3].

The first network consisted of 3410 web pages and was formed from the keywords `amusement parks`. Algebraic smoothing created 3383 blocks. Most blocks consisted of a single variable. More

Pointwise		Block	
$\|r\|$	Conv. Fac.	$\|r\|$	Conv. Fac.
125.69600	1.00000	125.69600	1.00000
1.22382	0.00974	8.51119	0.06771
0.90847	0.74233	2.72509	0.32018
0.79502	0.87511	1.27415	0.46756
0.70719	0.88954	0.76593	0.60113

TABLE 4.12

Single-element thick 2-D plane-stress cantilever beam discretized with rectangular elements on a 64×1 *grid. Stretched elements use a* $10:1$ *aspect ratio. For* $\alpha = 0.1$*, the algebraic block smoothing created 126 blocks.*

Pointwise		Block	
$\|r\|$	Conv. Fac.	$\|r\|$	Conv. Fac.
2.531849	1.00000	2.531849	1.00000
1.270078	0.50164	1.295928	0.51185
0.514680	0.40523	0.518006	0.39972
0.040481	0.07865	0.025487	0.04920
0.022789	0.56295	0.006628	0.26006
0.015676	0.68787	0.001816	0.27392
0.011209	0.71504	0.000513	0.28235
0.008079	0.72081	0.000149	0.29073
0.005834	0.72208	0.000043	0.29113
0.004214	0.72237	0.000013	0.29402
0.003045	0.72244	0.000004	0.29759

TABLE 4.13

Algebraic block smoothing applied to linear formulation of PageRank on a network with 3410 web pages. The algorithm chose 3383 blocks.

notably, given the improvement in convergence, the method chose 13, 4, and 2 blocks of size 2, 3, and 4, respectively. Such results, seen in Table 4.13 underscore the method's ability to judiciously select blocks. Moreover, a relatively small number of blocks contain more than a single variable but result in a marked decrease in the rate of convergence.

The second network consisted of 5354 web pages formed from the keyword `blues`. Algebraic smoothing created 5333. Again, most blocks consisted of 1 variable. This time 13, 2, and 1 blocks are formed of size 2, 3, and 5, respectively. Table 4.14 again reflects the increased efficiency resulting from the method.

We do not see the same level of speed-up on all networks. For instance, the network formed from the keywords `automobile industries` led to the results in Table 4.15. Note, only 4 blocks consisted of more than one variable. More specifically, 3 blocks consisted of two variables and one block consisted of 6 variables. Note that the residual is lower on the last iterate but the asymptotic convergence rate, while better, is not significant.

This section underlines that ability of the algebraic block smoothing method of this paper to be applied successfully to nonsymmetric linear systems.

5. Conclusions and Future Work. This paper presented a new method of adaptively and algebraically constructing smoothers based on LSA for multigrid methods. As presented, the method can be used in the context of both geometric and algebraic multigrid methods. Current research includes work on implementing LSA into traditional AMG for the purposes of widening the scope of problems it can solve effectively.

Numerical results of this paper reflect the seamless way the adaptive algebraic smoother can

Pointwise		Block	
$\|r\|$	Conv. Fac.	$\|r\|$	Conv. Fac.
Pointwise		Block	
$\|r\|$	Conv. Fac.	$\|r\|$	Conv. Fac.
2.45576	1.00000	2.45576	1.00000
0.69123	0.28147	0.71166	0.28979
0.07483	0.10826	0.02998	0.04212
0.04167	0.55683	0.00696	0.23208
0.02741	0.65785	0.00194	0.27946
0.01905	0.69485	0.00059	0.30205
0.01348	0.70761	0.00018	0.31228
0.00960	0.71249	0.00006	0.31750
0.00686	0.71486	0.00002	0.32013

TABLE 4.14

Algebraic block smoothing applied to linear formulation of PageRank on a network with 5354 web pages. The algorithm chose 5333 blocks.

Pointwise		Block	
$\|r\|$	Conv. Fac.	$\|r\|$	Conv. Fac.
2.57988	1.00000	2.57988	1.00000
0.12570	0.04872	0.12583	0.04877
0.01807	0.14374	0.01569	0.12468
0.00851	0.47116	0.00573	0.36527
0.00526	0.61831	0.00274	0.47777
0.00362	0.68833	0.00166	0.60456
0.00255	0.70478	0.00108	0.65079
0.00181	0.70921	0.00071	0.66034
0.00129	0.71148	0.00047	0.66209

TABLE 4.15

Algebraic block smoothing applied to linear formulation of PageRank on a network with 1196 web pages. The algorithm chose 1188 blocks.

transition between scalar and system PDEs and also how the method is effective on nonsymmetric linear systems that are not generated from PDEs. The adaptive algebraic method of this paper allows the smoothing process to adapt to various problem types and to algebraically design such smoothers to be more effective than pointwise Gauss-Seidel. The simplicity of the method will allow it to be easily incorporated into existing multigrid codes. The methods provide a powerful tool for adaptively constructing smoothers and can complement existing research on coarse-grid correction components.

There are several new directions in which the present work can be extended. Firstly, the close relationship between the strength of coupling measure presented in this paper and the coarsening measure presented in [6], indicate that it is worthwhile considering whether our measure can be used as a criteria for coarsening non-symmetric and coupled system PDE systems in AMG. Secondly, LSA provides a matrix reordering based on strength of coupling as opposed to alternatives based on minimizing fill-in or finding independent subsets. It is worthwhile to consider whether a matrix reordering scheme based on some combination of these schemes will offer better solver performance in a variety of applications. Thirdly, the adaptivity of the block iterative smoothers presented here are attractive in the context of fully adaptive multigrid methods. These are directions we are currently pursuing.

[**TODO**: Bobby, do we want to include anything on future work? For instance, we have completely removed the connection of the strong couplings as possibly serving as a replacement to AMG's strong

connection. We allude to it which may suffice. Tim, let me know what you think of the last para. We can keep mum on all or part of it]

6. Acknowledgements. The first author would like to thank Mac Hyman for introducing him to the topic of sensitivity analysis (SA) and thank both him and Leon Arriola for providing drafts of their book on sensitivity and uncertainty quantification. Using SA for developing smoothers is the main idea that this paper introduces. The first author would also like to acknowledge the funding provided by the LANL LDRD office under LDRD 20050315ER which made this research possible. The second author thanks the U.S. Department of Energy for partially funding this research through the U.S. Department of Energy grant DE-FG02-04ER25590. Both authors would like to thank Leon Arriola for pointing us to the expository article [2] on sensitivity analysis and for many useful and entertaining discussions. We would like to thank John Ruge for sharing with us his thoughts and initial attempts at developing ideas in the spirit of LSA for identifying strength in algebraic multigrid methods. The authors also thank Klaus Stüben for sharing the linear systems in Sections 4.1.3 and 4.1.4 and also for giving permission for the use of Figure 4.3 as it appeared in Section 8.4.1 of [25].

REFERENCES

[1] J. O. amd D. Szyld, *A block ordering method for sparse matrices.*, SIAM journal on scientific computing, 11 (1990), pp. 811 –.
[2] L. M. Arriola and J. M. Hyman, *Being sensitive to uncertainty*, Computing in Science and Engg., 9 (2007), pp. 10–20.
[3] A. Borodin, G. O. Roberts, P. S. Rosenthal, and P. Tsaparas, *Finding authorities and hubs from link structures on the world wide web*, in Proceedings of the 10th International World Wide Web Conference, May 2000.
[4] A. Brandt, *Multigrid Techniques: 1984 Guide with Applications to Fluid Dynamics*, 1984.
[5] A. Brandt, *General highly accurate algebraic coarsening schemes*, Electrical Transactions on Numerical Analysis, 10 (2000), pp. 1–20.
[6] J. Brannick, M. Brezina, S. MacLachlan, T. Manteuffel, S. McCormick, and J. Ruge, *An energy-based AMG coarsening strategy*, Numer. Linear Algebra Appl., 13 (2006), pp. 133–148.
[7] R. Bridson, *Multi-resolution approximate inverses*, 1999.
[8] O. Broker and M. J. Grote, *Sparse approximate inverse smoothers for geometric and algebraic multigrid.*, Applied numerical mathematics, 41 (2002), pp. 61 –.
[9] T. Chartier, R. D. Falgout, V. E. Henson, J. Jones, T. Manteuffel, S. McCormick, J. Ruge, and P. S. Vassilevski, *Spectral AMGe (ρAMGe)*, SIAM J. Sci. Comput., 25 (2003), pp. 1–26 (electronic).
[10] A. J. Cleary, R. D. Falgout, V. E. Henson, and J. E. Jones, *Coarse-grid selection for parallel algebraic multigrid*, in Solving irregularly structured problems in parallel (Berkeley, CA, 1998), vol. 1457 of Lecture Notes in Comput. Sci., Springer, Berlin, 1998, pp. 104–115.
[11] J. Cleary, R. Falgout, V. Henson, J. Jones, T. Manteuffel, S. McCormick, G. Miranda, and J. Ruge, *Robustness and scalability of algebraic multigrid*, SIAM Journal on Scientific Computing, 21 (2000), pp. 1886–1908.
[12] A. George, *An automatic nested dissection algorithm for irregular finite element problems.*, SIAM journal on numerical analysis, 15 (1978), pp. 1053 –.
[13] V. E. Henson and U. M. Yang, *BoomerAMG: a parallel algebraic multigrid solver and preconditioner*, Appl. Numer. Math., 41 (2002), pp. 155–177. Developments and trends in iterative methods for large systems of equations—in memoriam Rüdiger Weiss (Lausanne, 2000).
[14] J. Jones, *Multigrid methods for systems of PDEs.* Invited talk, March 2007. AMS 2007 Spring Southeastern Section Meeting, Davidson, NC, Special Session on Recent Applications of Numerical Linear Algebra.
[15] G. Karypsis, *A fast and highly quality multilevel scheme for partitioning irregular graphs.*, SIAM journal on scientific computing, 20 (1999), pp. 359 –.
[16] J. M. Kleinberg, *Authoritative sources in a hyperlinked environment*, J. ACM, 46 (1999), pp. 604–632.
[17] A. N. Langville and C. D. Meyer, *Google's PageRank and beyond: the science of search engine rankings*, Princeton University Press, Princeton, NJ, 2006.
[18] M. R. Leuze, *Independent set orderings for parallel matrix factorization by gaussian elimination.*, Parallel computing, 10 (1989), pp. 177 –.
[19] J. W. H. LIU, *Modification of the minimum-degree algorithm by multiple elimination.*, ACM transactions on mathematical software, 11 (1985), pp. 141 –.
[20] O. Livne, *Coarsening by compatible relaxation*, Numerical Linear Algebra with Applications, 11 (2004), pp. 205–227.
[21] G. Meurant, *A review on the inverse of symmetric tridiagonal and block tridiagonal matrices*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 707–728.

[22] S. V. Parter, *Multiline iterative methods for elliptic difference equations and fundamental frequencies*, Numerische Mathematik, 3 (1961), pp. 305 –.
[23] J. Ruge and K. Stüben, *Algebraic multigrid*, in Multigrid Methods, S. McCormick, ed., vol. 3 of Frontiers in Applied Mathematics, SIAM, Philadelphia, 1987, ch. 4, pp. 73–130.
[24] Y. Saad, *Arms: an algebraic recursive multilevel solver for general sparse linear systems.*, Numerical linear algebra with applications, 9 (2002), pp. 359 –.
[25] K. Stüben, *Algebraic multigrid (AMG): An introduction with applications*, Gesellschaft für Mathematik und Datenveranbeitung, Nr. 70, (1999).
[26] W. P. Tang and W. Wan, *Sparse approximate inverse smoother for multigrid.*, SIAM journal on matrix analysis and applications, 21 (2000), pp. 1236 –.
[27] U. Trottenberg, C. W. Oosterlee, and A. Schüller, *Multigrid*, Academic Press Inc., San Diego, CA, 2001. With contributions by A. Brandt, P. Oswald and K. Stüben.
[28] P. Tsaparas, *Data Sets for Link Analysis Ranking Experiments.* `http://www.cs.toronto.edu/~tsap/experiments/download/download.html`, accessed July 2007.
[29] ———, *Link Analysis Ranking*, PhD thesis, University of Toronto, 2004.
[30] R. S. Varga, *Matrix Iterative Analysis*, Springer Series in Computational Mathematics, San Diego, CA, 1999.
[31] I. Yavneh, *A method for devising efficient multigrid smoothers for complicated PDE systems*, SIAM J. Sci. Comput., 14 (1993), pp. 1437–1463.